Explaining Free and Open Source Software

by Scot Colford

Open Source Software in Libraries

Communicating the benefits and limitations of free and open source software to the less technically experienced can be (to say the least) challenging, so before attempting to do so, the smart professional will prepare himself or herself with three things:

- A thorough (but concise) description
- The ability to correct misconceptions
- A list of open source applications an organization can (and may already) benefit from.

What It Is

In the introductory technology course I teach for graduate LIS students at Simmons College, students usually grasp the concept of compiled software (vs. scripting) fairly easily, so I often approach the topic of open source as a metaphor.

> *Imagine that it's your job to buy a cake for a co-worker's birthday. You go to the bakery to see what they've got on display and you find a lovely white cake with a beautiful yellow icing with "Happy Birthday!" flowing across it. It's even kosher. But there's a problem. The frosting is pink and the birthday person just can't stand that color. There are other pre-made cakes available but they have even more things wrong with them, like the chocolate cake with "Sto Lat!" written on it. (That's Polish.) You can order a custom cake, of course, but it's more expensive, takes a week and you have your doubts that the bakery can meet your specific demands. What do you do?*

Scot Colford is web services manager at the Boston Public Library. He can be reached by email at scolford<at>bpl.org

> *Why don't you bake your own cake? It would be pretty hard if you had to guess at the ingredients and just experiment with various ratios of flour, butter and sugar. You might be better off with the pink cake from the bakery. But that's why Betty Crocker invented new recipes. It's a proven way to create a cake. (And a list of ingredients is freely redistributable [1] by the way.) You can fiddle with the recipe to your heart's content to make the exact cake you need. And when you're done, you can give the list of ingredients to anyone else facing a similar situation.*

Once the penny drops, I go on to explain that free and open source software is a lot like the cake that you made from the recipe. It's a creation that owes a lot to the person or people who created the original recipe, but since you were given the building blocks to create this end result, you had the opportunity to alter them to suit your needs. Ultimately, your modifications should help others who have needs similar to yours, and often, they might even be the people from whom you got the recipe in the first place. It's an evolutionary process that feeds back into itself.

Often, a student will then ask where open source software may be purchased. God bless him. I have heard colleagues ask the same thing. While the concept of direct access to application source code can be described in metaphor, the topic of licensing involves taking a look at licenses, those things we all like to click past in our rush to install software. It is necessary to broach the topic, however, since the license is truly what makes free software "free" and open source software "open." It is also useful to understand the difference between free software, as defined by the Free Software Foundation (FSF), and open source software, as defined by the Open Source Initiative (OSI).

The free software definition was published in 1986 by Richard Stallman,

NEXT PAGE >

president then and now of the FSF. The definition codifies four essential freedoms that computer software users should be entitled to:

- The freedom to run the program for any purpose.
- The freedom to study how the program works and adapt it to your needs.
- The freedom to redistribute copies so you can help your neighbor.
- The freedom to improve the program and release your improvements to the public, so that the whole community benefits [2].

With the emphasis on these four freedoms, free software is software that end users have freedom to alter, run and redistribute as they see fit. The label "free software" often causes confusion. "Free," of course, also carries the connotation of "without cost," when in reality, cost is not a criterion for free software. To address the ambiguity between "free as in free speech" versus "free as in free beer" mentioned in the free software definition, over the years a number of alternative terms have been suggested, mostly using non-English words that have unequivocal definitions. Software *libre*, for example, uses the Spanish and French adjective meaning "free" in the same sense as "liberty." Similarly, software licensed free of charge is sometimes labeled software *gratis*. While these (and similar) terms have been adopted in non-English speaking countries – and while the FSF officially supports any term conveying the concept of liberty – software *gratis* and software *libre* have not been widely adopted in the United States. Instead, cost-free software is generally labeled "freeware" and the main FSF-approved label for liberty-infused software in use is "free software."

This confusion, along with the confrontational activist stance of the FSF in defending these freedoms, led to the formation of the Open Source Initiative in 1998 and the open source definition [3]. Ten criteria must be met in order for a software distribution to be considered open source:

1. Free redistribution – The license must allow end users to redistribute the software, even as part of a larger software package and may not charge royalties for this right.
2. Source code – The distribution must make the source code freely available to developers.
3. Derived works – The license must permit modifications to be made to the software for redistribution under the same license.
4. Integrity of the author's source code – The license may require that modified distributions be renamed, or that modifications be made via patch files rather than modifying the source code.
5. No discrimination against persons or groups
6. No discrimination against fields of endeavor – This includes commercial or controversial endeavors.
7. Distribution of license – The same license must be passed on to others when the program is redistributed.
8. License must not be specific to a product – A program may be extracted from a larger distribution and used under the same license.
9. License must not restrict other software – The license cannot prescribe the terms of other software with which it is distributed.
10. License must be technology-neutral – The license cannot restrict the use of the program to any individual interface or platform [4].

While the Open Source Initiative has approved over 50 different licenses as meeting the criteria of the organization, a list of the nine most widely used licenses is a sufficient sample to get an overview of the different restrictions and freedoms they provide:

- Apache Software License 2.0 (www.apache.org/licenses/LICENSE-2.0.html)
- New BSD License (www.opensource.org/licenses/bsd-license.php)
- GNU General Public License (GPL) (www.gnu.org/licenses/gpl.html)
- GNU Lesser General Public License (LGPL) (www.gnu.org/licenses/lgpl.html)
- MIT License (www.opensource.org/licenses/mit-license.php)
- Mozilla Public License 1.1 (MPL) (www.mozilla.org/MPL/MPL-1.1.html)
- Common Development and Distribution License (www.sun.com/cddl/cddl.html)
- Common Public License 1.0 (www.ibm.com/developerworks/library/os-cpl.html)
- Eclipse Public License (www.eclipse.org/legal/epl-v10.html) [5].

Despite the few ideological differences between open source and free software, for practical purposes they provide the same basic advantages (and challenges) in a library or information science setting. For this reason, they are often referred to under a collective term such as "free and open source software," FOSS, F/OSS or other terms.

What It Is Not

A more difficult task than defining free and open source software for novices is combating the inevitable assumptions and misinformation that materialize. The most common misconception, alluded to above, is that since the source code is freely distributed without royalty or licensing fee, open source applications are free of cost. While it is possible for a library or other organization to avoid buying a proprietary software package, open source may carry a plethora of hidden costs in development and maintenances, particularly if any customization is to be made to the software. These costs may translate into salaries for additional technical staff or possibly external support, development and/or hosting services such as the consulting service LibLime.

In my experience, the staff most susceptible to the "free lunch" myth are overeager novice technicians or new librarians hoping to stretch dwindling budgets as far as possible. Veteran administrators, on the other hand, seem to regard free and open source solutions with suspicion. While some of their caution is of the "you get what you pay for" variety, the absence of an accountable vendor causes distress to some of the old library guard. Without a contract to point to, non-technical administrators seem to feel that development of library services is in the hands of an unknown middle-aged, unemployed social misfit coding in his parents' basement at three a.m. between reruns of *Stargate* and *Star Trek: Deep Space Nine*. It is an image closely associated with hacker culture and all the bugaboos associated with it, such as Bill Gates' attempt to frame FOSS advocates as "modern day Communists." [6]. It is a valid point – not the image of the basement coder, but rather that the people currently working on any given open source application are not doing so to win customers; they are attempting to solve specific problems.

Richard Stallman clearly illustrates this clever problem-solving ability of developers in his 2002 article "On Hacking." [7]. "Playfully doing something difficult" is hacking, according to Stallman, as opposed to the criminal connotation sometimes associated with the word. A hacker enjoys puzzles and finding efficient solutions to seemingly insurmountable problems, something that libraries and information centers seem to have no lack of. An organization must hire either staff or consultants with this special ability to contribute to the larger problem-solving community if its unique concerns are to be addressed in an open source application. It is an entirely new and exciting paradigm to invite these creative people into our space, rather than knowing they are sequestered out of reach behind a vendor's competitive gate.

Along the same lines, non-technical staff are often concerned about the perceived absence of technical support available for open source projects. Library vendors charge an incredible amount of money to support the software they license, typically in the form of yearly maintenance fees. A significant portion of a technology operating budget can be spent this way in exchange for the privilege of calling the vendor when the software fails (one hopes 24/7, but sometimes only 9-5 weekdays), when a bug is identified (that perhaps the vendor has already documented but about which has not thought to inform customers) or when documentation proves incorrect or inadequate (sometimes referred to as a "training issue" by vendors). On occasion, the maintenance fee even entitles customers to conversations such as this one.

Me: ... So, the end result is that the online catalog informs all our patrons that they will be notified by phone when their holds arrive.

Vendor Representative: Okay.

Me: Even if the individual patron will actually be notified by email or snail mail.

Vendor Rep: I see. But the default notification method for the location is "phone."

Me: I know, but the patron's choice overrides this when the notice is sent.

Vendor Rep: Yes. But this is the way it was designed to work.

Me: Incorrectly?

Vendor Rep: Well, you're certainly welcome to fill out an enhancement request...

While not all vendor support experiences are as fruitless as the tongue-in-cheek examples above, technical staff will recognize that the promise of support from a proprietary software vendor rarely matches value – monetary or quality – attributed to it by the vendor or non-technical library staff. Support from a hardware vendor can be exemplary without much effort. If a component breaks, swap it out with a working piece. Software packages, however, are complex systems that usually do not function in such a modular fashion. Well, not in proprietary applications, at any rate.

Free and open source software application users, on the other hand, must rely on development communities for support. Users and developers produce documentation, write installation guides and answer specific support questions in forums, not because they are bound by contract, but because they can learn more about the software that they, too, are using. One can see this type of activity in proprietary library software user groups as well. Indeed, many systems librarians around the world find user group listservs much more illuminating about how an application works than vendor-supplied documentation or training. Many systems librarians also spend a large amount of time writing scripts, developing external applications or finding unintended creative uses for application features. Vendors sometimes even celebrate these accomplishments at annual user group conferences. That recognition is nice, but the upshot of this type of grassroots support of proprietary software is summarized most succinctly by a meme I hear frequently repeated by my colleague Michael Klein: "The workaround has become the work." When practical issues with open system software are addressed by those using the software, however, the solutions can be immediately returned to the user community as a new release. A user's contribution is not just a workaround. It is the essential work. No vendors are needed and it will not matter if you missed the user group conference.

Misconceptions such as those mentioned above can transform into terrifying specters sure to doom the success of any open source project at an organization. Nevertheless, a well-prepared technical staff member at a library or information-centric organization can circumvent misunderstandings and turn stakeholder anxiety into excitement, provided that an open source alternative is truly the best option. A full cost benefit analysis should be performed taking in account all of the factors mentioned above for both proprietary and open source alternatives, including the following:

- Licensing
- Recurring fees, such as maintenance
- Personnel costs for development and maintenance cycles
- Amount and time of additional *development* required for missing features
- Amount and time of *workarounds* required for missing features
- The benefit of contributing to the support community
- The "lock in" aspect of committing to a proprietary model
- The ease with which one can (or can't) migrate to a new platform, if necessary.

If after this analysis, free or open source software seems to offer significantly more benefits, developing a quick fact sheet for administrators comparing a specific FOSS application to a known proprietary equivalent can quickly impress. Above all, a functional prototype created in the FOSS application will dispel concerns that the software is somehow rudimentary or experimental. Another oft-repeated meme among my colleagues is "working code works," and it is true. Nothing illustrates your point better than illustrating your point.

You're Soaking In It

If conversations with non-technical staff veer off into the uncomfortable realm of doubt and uncertainty, they may start asking questions like "Are we sure that we are ready to *invest* in open source software?" or "Do you think we have investigated enough to *commit* to open source?" This moment is an excellent opportunity to pull out your best Madge, the Palmolive manicurist impression, and quip, "Commit to open source? Why, you're soaking in it!"

The pervasiveness of the World Wide Web guarantees that nearly every information organization is using free or open source software to perform some function. For example, 43.7% of web browsing is being done with Firefox, an open source application, and Internet Explorer is steadily losing its lead. It only has 50.5% of the market currently [8]. Similarly, 49.82% of web servers are running Apache, which has retained its first-place spot over Microsoft IIS for 12 years [9]. It seems that every month, another visible

TOP OF ARTICLE
< PREVIOUS PAGE
NEXT PAGE >

library announces its website redesign in Drupal. And if an organization hosts a blog or a wiki, the chances that it is an open source package are pretty good. In fact, the chance that your organization's hosted blog is powered by WordPress is pretty good simply because it is supported by one of the most active open source communities in cyberspace.

Widely used open source applications	
Operating Systems	**Web and Proxy Servers**
GNU/Linux	Apache
FreeBSD/OpenBSD	Squid cache
Web Browsers	**Blogs**
Firefox	WordPress
Lynx	Movable Type
Office Software	**Wikis**
OpenOffice.org	MediaWiki
PDFCreator	TikiWiki
Instant Messengers	**Content Management Systems**
Gaim	Drupal
Pidgin	Joomia

Realizing that your organization is already a hybrid environment helps administrators and staff realize that a relationship with open source can be more like a respectful, close friendship than a toxic, codependent marriage. Open source will let you develop relationships with other software packages, open or proprietary. It is true that there are some great philosophical justifications for using FOSS. Just as many of us cannot ride a bicycle to every destination and instead opt to buy a hybrid car as a compromise, one can consider those noble justifications while being "as open as possible." ("AOAP" is the meme, as Mr. Klein reminds me.)

Open source library applications	
ILSs	**OPACs**
Evergreen	Blacklight
Koha	Fac-Back-OPAC
Repositories	MARC Module for Drupal
Digital Asset Factory	Scriblio
DSpace	SOPAC
Fedora	VuFind
Metasearch Resolvers	
CUFTS	
LibraryFind	

A commitment need only be as deep as required by the organization and exploration can be done without an intention to replace proprietary software currently in place. Much has been written about the open source integrated library systems Koha and Evergreen, but if an organization currently has an ILS in place, it may still be worthwhile to install an alternate web OPAC like Scriblio, SOPAC or VuFind instead. The installation can live alongside the current system and, if presented as a public beta, can provide useful data regarding what users prefer from a catalog interface. There are hundreds of library applications in development, though it is advantageous to choose a project with an active development community.

Conclusion

Open source software can unnerve staff and administrators who do not have a full understanding of the concept, the myths and the all-around usefulness of it. Developers and technical staff who can communicate these three things will find it much easier to integrate some truly innovative software into their organization's technical environment. ■

Resources Cited in the Article

[1] U.S. Copyright Office (2008). *Recipes.* Retrieved August 24, 2008, from www.copyright.gov/fls/fl122.html
[2] Free Software Foundation (2007). *The free software definition.* Retrieved August 24, 2008, from www.fsf.org/licensing/essays/free-sw.html
[3] Tiemann, M. (2006). *History of the OSI.* Retrieved August 24, 2008 from www.opensource.org/history/
[4] Coar, K. (2006). *The open source definition.* Retrieved August 24, 2008, from www.opensource.org/docs/osd/
[5] Nelson, R. (2006). *Open source licenses by category.* Retrieved August 24, 2008, from www.opensource.org/licenses/category/
[6] Brown, A. (2005, January 11). The war on copyright communists: Bill Gates wants software patents to protect his profit, not the public. *The Guardian [London, England]*, p.22.
[7] Retrieved August 24, 2008 from www.stallman.org/articles/on-hacking.html
[8] W3Schools. (2008). *Browser statistics.* Retrieved September 1, 2008, from www.w3schools.com/browsers/browsers_stats.asp
[9] Netcraft, Ltd. (2008). *August 2008 Web server survey.* Retrieved September 1, 2008, from http://news.netcraft.com/archives/2008/08/29/august_2008_web_server_survey.html

TOP OF ARTICLE
< PREVIOUS PAGE